Courant Computer Science Report #5

October 1974

Investigations in the Theory of Descriptive Complexity

William L. Gewirtz

Courant Institute of Mathematical Sciences

Computer Science Department

New York University

Report No. NSO-5 prepared under Grant No. NSF-GJ-31363x1 from the National Science Foundation

COURANT INSTITUTE OF MATHEMATICAL SCIENCES

Computer Science NSO-5

INVESTIGATIONS IN THE THEORY OF DESCRIPTIVE COMPLEXITY

William L. Gewirtz

Report No. NSO-5 prepared under
Grant No. NSF-GJ-31363X1 from
the National Science Foundation

TABLE OF CONENTS

Page

INTRODUCTION 1

CHAPTER

1. Fundamental Notations, Definitions and Results of Descriptive Complexity 4
   - 1.1 Notational Conventions 4
   - 1.2 Fundamental Definitions and Results of the Kolmogorov[1] Formulations 5
   - 1.3 Chaitin's Formulation of Descriptive Complexity 10
   - 1.4 Elementary Theorems Relating the Kolmogorov and Chaitin Formulations 17
   - 1.5 Recursive and Recursively Enumerable Sequences 21
2. INFINITE RANDOM SEQUENCES 23
   - 2.1 Basic Definitions and Results of the Kolmogorov Formulation 23
   - 2.2 Chaitin's Proposed Definitions 24
   - 2.3 A Definition Based on Chaitin's Formulation 26
   - 2.4 Recursive and Recursively Enumerable Subsequences 28
3. AN ANALYSIS OF $K(s^*/s)$ 31
   - 3.1 The Set of Optimal Programs 31
   - 3.2 $K(s^*/s)$ 32
4. ω_U - THE PROBABILITY THAT A UNIVERSAL COMPUTER IN THE SENSE OF CHAITIN HALTS 36
5. SUBRECURSIVE FORMULATIONS OF DESCRIPTIVE COMPLEXITY 40
   - 5.1 Introduction and Fundamental Definitions 40
   - 5.2 $K(s^{*j}/s)$ and Related Results 46
   - 5.3 A Hierarchy of Random Sequences 48
   - 5.4 The Maximum Grzegorczyk Level of All Programs of Length Less than n 50
   - 5.5 The Longest String of Complexity Less than n 53

NOTES 57

BIBLIOGRAPHY 59

Abstract

We consider various formulations of the notion of descriptive complexity. We obtain results relating different proposed definitions of an infinite random binary string. We prove that the set of optimal programs is immune and as a consequence obtain that the descriptive complexity of a program given what it computes is dependent on which formulation of descriptive complexity one is considering. We prove that in a certain strict sense for one formulation of descriptive complexity there is no optimal universal computer. We then develop the notion of descriptive complexity for a subrecursive formalism.

INTRODUCTION

This thesis will consider an assortment of problems in the area of descriptive complexity theory. We view a computer as an algorithm for a partial recursive function which reads a binary string input and then may or may not print another binary string which is its output. Relative to a particular computer C, the descriptive complexity of a particular string s is defined as the length of the shortest string p which when inputted to C will cause C to output s. This notion has been formalized in a number of different ways. We consider three:

1. Kolmogorov and Chaitin's <u>absolute complexity</u> in which C with input p must output s given no additional information.

2. Kolmogorov and Chaitin's <u>conditional complexity</u> in which C must output s given p and information about the length of s.

3. Chaitin's recent definition in which C must output s given a program p which is <u>self-delimiting</u>.

Though Chaitin and Kolmogorov were co-founders of the notions of descriptive complexity as outlined in 1. and 2. above we will, for expository purposes, refer only to Kolmogorov in order to distinguish these earlier formulations from Chaitin's recent formulation.

We also consider the relative complexity of s given t by which we mean the length of the shortest program p which will cause C to output s given information about t. Different formalisms arise from considering various different ways in which t can be given, i.e. either by assuming t itself is given or by assuming a string t' from which t can be computed is given.

In terms of each of the three above approaches a random or patternless string s is defined as one whose shortest program is of length essentially equal to itself. Intuitively, if a string contains some pattern then that pattern can be exploited by the computer in outputting the string. However, if a string s is genuinely patternless the shortest program for outputting it might consist of inputting s and instructing the computer to output its input.

Chapter 1. will present most of the fundamental notations, definitions and results developed for the most part by others which are relevant to our work.

Chapter 2. will consider five definitions of the notion of infinite random sequence. Two have been previously proposed and subsequently proven equivalent. Two have been proposed by Chaitin. We show that one of these is at least as restrictive as the other. We then propose a new definition based on the Chaitin definition of descriptive complexity which we prove is at least as restrictive as the first two.

Chapter 3. considers the set of optimal (shortest) programs proving that this set is immune. We then consider the relative complexity of an optimal program given what it computes. We prove that with respect to this question Chaitin's definition of complexity differs sharply from Kolmogorov's.

Chapter 4. deals entirely with a result based on Chaitin's formulation: we prove that given any optimal universal computer there exists another optimal universal computer for which the complexity of all sufficiently long binary strings is uniformly reduced.

Chapter 5. deals with various definitions of descriptive complexity where the function defined by the computer C is assumed <u>subrecursive</u>. Various results are obtained and comparisons with the previous work are made. The reader should note that in Chapter 5. we will state and use many results concerning our new subrecursive formulations without proof since it should be evident that the proofs of these results follow the same lines of reasoning used to prove corresponding results earlier obtained for the full recursive formulations.

Chapter 1

FUNDAMENTAL NOTATIONS, DEFINITIONS AND RESULTS OF DESCRIPTIVE COMPLEXITY

1.1. Notational Conventions

Let $X = \{0,1\}$

$X^* = \{$strings of finite length over $X\}$

$X^\infty = \{$infinite strings over $X\}$

$\Lambda =$ empty string

$N = \{$set of natural numbers$\}$.

u,v,w,x,y,z	will denote members of X^∞.
p,q,r,s,t	will denote members of X^*.
i,j,k,ℓ,m,n	will denote members of N.
α,β,γ	will denote members of $X^* \cup X^\infty$.

If $P(n)$ is a predicate then $\overset{\infty}{\exists} n P(n)$ will mean that there exist infinitely many n for which $P(n)$ is true. For any string $s \in X^*$, $|s|$ will denote its length (e.g. $|\Lambda| = 0$; $|0100| = 4$).

For any string $\alpha \in X^* \cup X^\infty$, α^n will denote the initial segment of α of length n. If $n > |\alpha|$ then $\alpha = \alpha$. Also α_n will denote the nth bit of α.

X^* is assumed ordered in the conventional way; that is, $s < t$ if either $|s| < |t|$, or $|s| = |t|$ and s precedes t in the lexocographic ordering with $0 < 1$. We associate the integer n with the nth element of X^*. By $\dot{n}$ we will mean the length of the string associated with n.

If $s = s_1 s_2 \ldots s_n$ then by $\bar{s}$ we denote the string $s_1 s_1 s_2 s_2 \ldots s_n s_n 01$. By $<s,t>$ we denote the string $\bar{s}t$ from which both s and t can be unambiguously recovered. We also denote by $<\dot{n},t>$ the string $\bar{q}t$ where q is the string associated with n. Obviously $|<s,t>| = 2|s| + |t| + 2$.

When referring to a measure on X^∞ we mean the usual product measure on X^∞ relative to the probabilities $\frac{1}{2}$ for 0 and 1.

1.2. Fundamental Definitions and Results of the Kolmogorov[1] Formulations

We recall that by a "computer" A we mean any algorithm for a partial recursive function.

Def. 1.2.1. (<u>Absolute Descriptive Complexity</u>)[2]

Let A be a computer which maps $X^* \to X^*$ then:

$$K_A(s) = \left\{\begin{array}{ll} \min\limits_{A(p) = s} |p| & \\ \infty & \text{if } \forall p(A(p) \neq s). \end{array}\right\}$$

If one chooses to think of A as a stored program computer then p should be thought of as program + data. If one thinks of A as a particular Turing machine then p is the initial tape content.

It has been established[3] that there exist universal machines U which are optimal in the following sense:

$$\forall s[K_U(s) \leq K_A(s) + c_{A,U}].$$

In other words the complexity of any string s is at most $c_{A,U}$ bits longer measured relative to U than measured relative to A where the constant $c_{A,U}$ is independent of the particular string s and depends only on A and U.

Given $A_0, A_1, \ldots, A_n, \ldots$ some particular effective listing of all "computers," then relative to this particular listing we can define a universal computer U by the requirement:

$$\forall p[U(o^i 1p) = A_i(p)]$$

then clearly:

$$\forall s[K_U(s) \leq K_{A_i}(s) + (i+1)].$$

If U_1 and U_2 are two universal computers then:

$$\forall s[K_{U_1}(s) \leq K_{U_2}(s) + c_1 \;\&\; K_{U_2}(s) \leq K_{U_1}(s) + c_2]$$

and hence:

$$\forall s[|K_{U_1}(s) - K_{U_2}(s)| \leq c] \quad \text{where} \quad c = \max\{c_1, c_2\}$$

We can therefore choose a particular universal computer U and define $K(s) = K_U(s)$, and then read all of our results as accurate to within a constant.

Since there exists a particular computer A' (the identity computer) which outputs its input we have:

$$K_{A'}(s) = |s|$$

and since

$$K_U(s) \leq K_{A'}(s) + c$$

we conclude that

$$K(s) \leq |s| + c.$$

By $K(n)$ we mean $K(t)$ where t is the string associated with n.

Intuitively a string s is random if $K(s)$ is approximately $|s|$. We now define the notion of randomness for a finite string in terms of a parameter c which can be chosen arbitrarily as a positive integer.

Def. 1.2.2. A string s is random if $K(s) > |s| - c$.

THEOREM 1.2.1.[4] The number of strings of length n which are random is at least $(1 - 2^{-c+1})2^n$.

Proof: By a simple counting argument there are 2^n strings of length n and only 2^{n-c+1} programs of length less than or equal to $n-c$. Hence at least

$(2^n - 2^{-c+1})$ strings of length n are random.

Q.E.D.

If $K(s) = n$ then there exists at least one program p such that $|p| = n$ and $U(p) = s$. We denote the set of such programs by S^*. S^* has at least one element but may have more.

Def. 1.2.3. $s^* = \min\{t | t \in S^*\}$. (Here min refers to our assumed ordering of X^*.)

Def. 1.2.4. $s^{**} = \min\{t | t \in S^*$ and for all $r \in S^*$ the number of steps in the computation of $U(t)$ is less than or equal to number of steps[5] in the computation of $U(r)$.$\}$

Both $s \to s^*$ and $s \to s^{**}$ associate a unique program with the string s.

We now define the relative complexity of s given t. Here we assume that the underlying computer is an algorithm for any partial recursive function of two arguments mapping $X^* \times X^*$ into X^*.

Def. 1.2.5.

$$K_A(s/t) = \left\{\begin{matrix} \min |p| & \\ A(p,t) = s & \\ \infty & \text{if } \forall p(A(p,t) \neq s) \end{matrix}\right\}.$$

As before it has been shown[6] that there exists an optimal universal computer U for which:

$$\forall s \forall t [K_U(s/t) \leq K_A(s/t) + c_{A,U}].$$

Also if U_1 and U_2 are two universal computers then:

$$\forall s \forall t [|K_{U_1}(s/t) - K_{U_2}(s/t)| < c_{U_1,U_2}].$$

We then pick a particular universal computer U and define:

$$K(s/t) = K_U(s/t).$$

We also have as before:

$$K(s/t) \leq |s| + c.$$

Without loss of generality we will assume a single universal computer U which is universal for the absolute and relative definitions of complexity simultaneously.

Of particular importance is what Kolmogorov calls the conditional complexity of s: $K(s/|s|)$. [Note that $U(t,n)$ is defined as $U(t,n')$ where n' is the binary string associated with n].

Def. 1.2.6. A string is <u>conditionally</u> <u>random</u> if $K(s/|s|) > |s| - c$.

By the same counting argument used to prove Theorem 1.2.1 we have:

THEOREM 1.2.2.[7] The number of strings of length n which are conditionally random is at least $(1 - 2^{-c+1})2^n$.

THEOREM 1.2.3.[8] $K(s/|s|) \leq K(s) + c$.

Proof: Consider a computer C which given inputs q,r satisfies:

$$C(q,r) = U(q).$$

Then $K_C(s/|s|) = K(s)$ and since $K(s/|s|) \leq K_C(s/|s|) + c$ we conclude:

$$K(s/|s|) \leq K(s) + c.$$

Q.E.D.

1.3. Chaitin's Formulation of Descriptive Complexity

Chaitin[9] has proposed a different approach to the definition of descriptive complexity. His approach is best explained by considering a Turing model of computation in which there are two tapes. The computation begins with the program p inscribed on the input tape and the reading head scanning the first square of p. The work tape is initially blank if one is dealing with the absolute complexity and contains information about t if the computation is relative to t. The output is found on the work tape.

We now give a concrete definition of descriptive complexity couched in terms of a Turing model of computation. [An equivalent abstract definition which is not restricted to only a Turing model is given by Chaitin[10].]

Def. 1.3.1. Let A be a particular Turing machine

$$H_A(s) = \begin{cases} \min p \\ A(p) = s \text{ and when } A \text{ has halted it} \\ \text{is scanning the last square of the} \\ \text{input with the reading head never} \\ \text{having scanned a blank square.} \\ \infty \quad \text{if } \forall p(A(p) \neq s) \end{cases}$$

Loosely stated, Chaitin's definition requires that any program p be self-delimiting. The reasonableness of this definition derives from the following two considerations. First, it can be argued that the Kolmogorov definition of complexity allows for a hidden factor which uniformly decreases the complexity of any sequence. That is, when U computes s from p it ipso facto has available to it information not contained in p since it is able to determine the length of p by scanning the input till it encounters a blank square. Chaitin's definition essentially requires that such information be contained in p itself. Second, on a technical level the Kolmogorov definition does not have a subadditivity property that one tacitly assumes when programming with subroutines. For example, let s be the string $r \cdot t$, and let r^* and t^* be the minimal programs for r and t respectively. One would like to have a result of the type:

$$K(s) \leq K(r) + K(t) + c = |r^*| + |t^*| + c$$

(where c is independent of r,s and t.) Under the

THEOREM 1.3.1. a) There exists a universal optimal computer U in the sense that $\exists c \forall s [H_U(s) \leq H_A(s) + c]$ where c depends only on A and U.

b) For any two universal computers U_1, U_2: $\exists c \forall s [|H_{U_1}(s) - H_{U_2}(s)| < c]$ where c depends only on U_1 and U_2.

Hence we choose a particular universal computer U and define: $H(s) = H_U(s)$.

Def. 1.3.3. $s^* = \min p(U(p) = s)$.

We now define the relative complexity of s given t again in terms of a Turing model of computation. We recall that $A(p,n) = t$ means that Turing machine A with p inscribed on its input tape and r on its work tape halts with t inscribed on its work tape and the reading head on the input tape is scanning the last square of the input p, never having left the original input.

Def. 1.3.4.

$$H_A(s/t) = \left\{\begin{matrix} \min |p| & \\ A(p,t^*) = s & \\ \infty & \text{if } \forall p(A(p,t^*) \neq s. \end{matrix}\right\}$$

As before we get an analogue to Theorem 1.3.1 and thereby define $H(s/t) = H_U(s/t)$.

THEOREM 1.3.2. For any string t if $C(p,t)$ is defined then for all strings $r \neq \Lambda$ $C(p \cdot r,t)$ is undefined.

COROLLARY: For any string t, $S = \{p | C(p,t)$ is defined$\}$ is an instantaneous code.

We now define "probabilities" associated with a string s.

Def. 1.3.5.

$$P_C(s) = \sum_{C(p)=s} \frac{1}{2^{|p|}}$$

$$P_C(s/t) = \sum_{C(p,t^*)=s} \frac{1}{2^{|p|}}$$

$$P(s) = P_U(s)^{13}$$

$$P(s/t) = P_U(s/t)^{13}$$

THEOREM 1.3.3.

a) $\forall s [s^* \neq \Lambda]$

b) $0 \leq P_C(s) \leq 1$

c) $0 \leq P_C(s/t) \leq 1$

d) $H(s) \neq \infty$

e) $H(s/t) \neq \infty$

f) $P_C(s) \geq 2^{-H_C(s)}$

g) $P_C(s/t) \geq 2^{-H_C(s/t)}$

h) $0 < P(s) < 1$

i) $0 < P(s/t) < 1$

THEOREM 1.3.4.

a) For any computer C;

$$1 \geq \sum_{s} P\ (s) = \sum_{C(p)\text{defined}} \frac{1}{2^{|p|}}$$

b) For any computer C and string t

$$1 \geq \sum_{s} P\ (s/t) = \sum_{C(p,t^*)\text{defined}} \frac{1}{2^{|p|}}$$

Def. 1.3.6.

$$\omega = \omega_U = \sum_{s} P(s) = \sum_{U(p)\text{defined}} \frac{1}{2^{|p|}}$$

(ω is just the probability that a computer U halts).

THEOREM 1.3.5. There exists a c such that for all s:

a) $H(s/s) \leq c$

b) $H(H(s)/s) \leq c$

c) $H(s^*/s) \leq c$.

Def. 1.3.7. $<s_k,n_k>$ $(k = 0,1,2,\ldots)$ is a <u>list of satisfiable requirements</u> iff

1) $\sum_{k=0}^{\infty} \frac{1}{2^{n_k}} \leq 1$

2) For all k, $n_k > 0$.

3) There exists a recursive function $F\colon N \to X^*$ with infinite range such that $F(k) = <s_k,n_k>$.

C is said to satisfy a list of satisfiable requirements iff

1) For each $<s_k, n_k>$ there exists precisely one program p_k such that $|p_k| = n_k$ & $C(p_k) = s_k$.
2) For each p such that $C(p) = s$ there exist a k such that $|p| = n_k$ and $s = s_k$.

THEOREM 1.3.6. For any list of satisfiable requirements there exists a computer C which satisfies them.

By $H(n)$ we mean $H(t)$ where t is the string associated with n.

THEOREM 1.3.7. a) $\sum_{n=1}^{\infty} \frac{1}{2^{H(n)}}$ converges

For any recursive function F: $N \to N$:

b) If $\sum_{n=1}^{\infty} \frac{1}{2^{F(n)}}$ diverges (e.g. $F(n) = |n|$) then $H(n) > F(n)$ infinitely often.

c) If $\sum_{n=1}^{\infty} \frac{1}{2^{F(n)}}$ converges (e.g. $F(n) = (1 + \varepsilon)|n|$) there exist a c such that $H(n) < F(n) + c$.

THEOREM 1.3.8. a) There exist a c such that for all n if $|s| = n$ then $\max(H(s)) = n + H(n) + c$.

b) There exists a c such that for all n and k there are less than 2^{n-k+c} strings s of length n such that $H(s) \leq n + H(n) - k$.

C) There exists a c such that for all s and t:

t:

$$H(s) \leq H(s/t) + H(t) + c.$$

d) There exists a c such that for all n if $|s| = n$ then:

$$H(s) = H(s/n) + H(n) + c.$$

1.4. Elementary Theorems Relating the Kolmogorov and Chaitin Formulations.

THEOREM 1.4.1. if $K(s) = n$ then there exist a c such that for all s:

$$H(s) \leq n + 2|n| + c.$$

Proof: If $K(s) = n$ then there exists a program p such that $U(p) = s$ with $|p| = n$.

We now construct a computer C such that:

$$H_C(s) \leq n + 2|n| + 2.$$

C computes as follows on any input of the form:
$\lceil\lceil p \rceil\rceil \cdot p = \lceil n \rceil \cdot p$:

1. C first determines $|p|$ moving its reading head to the last square of $\lceil\lceil p \rceil\rceil$.
2. C then copies the next $|p|$ symbols onto its work tape.
3. C then simulates $U(p)$ without ever moving the reading head on the input tape from where it

was at the end of step 2.

It is therefore clear that if $K(s) = n$ then $H_C(s) \leq n + \rceil n \lceil = n + 2|n| + 2$. Hence since $H(s) \leq H_C(s) + c$ we conclude that $K(s) = n$ implies $H(s) \leq n + 2|n| + (c + 2)$.

Q.E.D.

COROLLARY: If $K(s) = n$ then $H(s) \leq n + 2||n|| + |n| + c \leq n + (1 + \varepsilon)|n| + c$.

Proof: By an argument similar to the one used in the above theorem we can construct a computer C for which

$$C(\rceil\rceil n \lceil\lceil \cdot |n| \cdot p) = U(p) \quad \text{where} \quad |p| = n.$$

COROLLARY: If $K(s) = n$ then $H(s) \leq n + H(n) + c$.

Proof: If $H(n) = k$ then there exists a program q such that $|q| = k$ and $U(q) = n'$ where n' is the string associated with the integer n. By an argument similar to the one used in the above theorem we can construct a computer C such that:

$$C(q \cdot p) = U(p).$$

THEOREM 1.4.2. Let s^* be a minimal program for s (either in the sense of Kolmogorov or Chaitin) then there exists a c such that for all s and t:

$$K(t/s^*) \leq K(t/s) + c.$$

Proof: There exists a computer C which on inputs p

and s^* computes $U(p,U(s^*))$. Hence

$$K_C(t/s^*) = K(t/s)$$

from which we conclude

$$K(t/s^*) \leq K(t/s) + c.$$

THEOREM 1.4.3. If $K(s/t^*) = n$ then there exists a c such that for all s and t:

a) $H(s/t) \leq n + 2|n| + c$

b) $H(s/t) \leq n + 2||n|| + |n| + c \leq n + (1 + \varepsilon)|n| + c$

c) $H(s/t) \leq n + H(n) + c$.

Proof: The proof of the above theorem follows the same lines as the arguments used to prove Theorem 1.4.1 and its corollaries.

We now wish to define $\tilde{H}(s/t)$ as a cross between the Kolmogorov and Chaitin formulations. That is, we will require self-deliniation but assume that U is given t and not t^*. Recalling the discussion prior to Definition 1.3.4 we have:

Def. 1.4.1.

$$\tilde{H}_A(s/t) = \left\{\begin{array}{ll} \min |p| & \\ A(p,t) = s & \\ \infty & \text{if } \forall p(A(p,t) \neq s). \end{array}\right\}$$

As before we get an analogue to Theorem 1.3.1 and thereby define $\tilde{H}(s/t) = \tilde{H}_U(s/t)$.

THEOREM 1.4.4. There exists a constant c such that for all s and t:

$$\tilde{H}(s/t) \geq H(s/t) - c.$$

Proof: Consider a computer C defined by

$$C(p,t^*) = U(p,U(t^*)).$$

Then

$$H_C(s/t) \leq \tilde{H}(s/t) + c,$$

and since

$$H(s/t) \leq H_C(s/t) + c$$

we conclude

$$H(s/t) \leq \tilde{H}(s/t) + c.$$

Q.E.D.

THEOREM 1.4.5. There exists a c independent of s and t such that:

$$\tilde{H}(s/t) \leq H(s/t) + H(t) + c.$$

Proof: The theorem follows immediately from $\tilde{H}(s/t) \leq H(s) + c$ and Theorem 1.3.8(c).

THEOREM 1.4.6. If $K(s/t) = n$ then there exists a c such that for all s and t:

a) $\tilde{H}(s/t) \leq n + 2|n| + c$

b) $\tilde{H}(s/t) \leq n + 2||n|| + |n| + c \leq n + (1 + \varepsilon)|n| + c$

c) $\tilde{H}(s/t) \leq n + H(n) + c.$

Proof: Theorem 1.4.6 is an exact analogue of Theorem 1.4.3.

1.5. Recursive and Recursively Enumerable Sequences.

Of special interest is the infinite sequence x defined by $x_i = 1$ if and only if $i \in R$ where R is usually a recursive or recursively enumerable set. x is then called the characteristic sequence associated with R. It is obvious that:

THEOREM 1.5.1.[14] If x is a characteristic sequence of any recursive set R, then $K(x^n/n) \leq c$.

If x is the characteristic sequence associated with a recursively enumerable set R then:

THEOREM 1.5.2.[15] a) $\exists c \forall n [K(x^n/n) \leq |n| + c]$

b) $\exists c \forall n [H(x^n/n) \leq \max H(i) \ (i \leq n)$.

Proof: Since R is recursively enumerable there exists a recursive function f which enumerates R. Let i be the last member of R less than n enumerated by f. In order to determine x^n we need only encode the function f and the integer i, the former requiring some constant number of bits.

Q.E.D.

The following shows that the bounds in the previous theorem were tight.

THEOREM 1.5.3.[15] There exists a recursively enumerable

R such that if x is its characteristic sequence then:

$$\exists c \forall n [K(x^n/n) > |n| - c].$$

Chapter 2

INFINITE RANDOM SEQUENCES

2.1. Basic Definitions and Results of the Kolmogorov Formulation.

We now consider the problem of defining an infinite random sequence. One might first attempt to define x to be random if and only if $\exists c \forall n [K(x^n) > n - c]$. This however has been shown by Martin-Löf to be incorrect since he has proven that no sequence x satisfies so strong a requirement. Before noting Martin-Löf's result, we quote a weaker result in an exercise in Feller's textbook[1] which implies that the set of x satisfying the above requirement is a set of measure 0.

THEOREM: Let N_n^x be the number of successive zeros ending in position n of the sequence x. Then with probability 1:

$$\overline{\lim_{n\to\infty}} \frac{N_n^x}{\text{Log}_2(n)} = 1.$$

By the above theorem for almost all strings x there exist infinitely many n for which

$$x^n \simeq x^{n-[\text{Log}_2(n)]} 0^{[\text{Log}_2(n)]}$$

(where $[r]$ denotes the integer part of r). For any such n it is clear that $K(x^n) \simeq n - \text{Log}_2(n)$. In fact the following theorem of Martin-Löf holds:

THEOREM:[2] Let $F(n)$ be any function for which $\sum_{i=1}^{\infty} \frac{1}{2^{F(i)}} = \infty$ (e.g. $F(n) = \mathrm{Log}_2(n)$). Then:

$$\forall x \overset{\infty}{\exists} n [K(x^n) < n - F(n)].$$

In light of Martin-Löf's result the following definitions of an infinite random sequence were proposed.[3]

Def. 2.1.1. x is <u>random(1)</u> iff $\exists c \overset{\infty}{\exists} n [K(x\ ) > n - c]$.

Def. 2.1.2. x is <u>random(2)</u> iff $\exists c \overset{\infty}{\exists} n [K(x\ /n) > n - c]$.

Daley[4] has shown that x is random(1) if and only if x is random(2). We will, therefore, refer to a sequence as <u>K-random</u> if and only if it is either random(1) or random(2).

Martin-Löf[5] and others have shown that $C = \{x | x \text{ is K-random}\}$ is a set of measure 1.

2.2. Chaitin's Proposed Definitions.

In terms of his formulation of the notion of complexity, Chaitin gives the following definition of an infinite random sequence.

Def. 2.2.1.[6] x is <u>C-random</u> iff $\exists c \forall n [H(x^n) > n - c]$.

This definition of Chaitin's relates to another definition of randomness Chaitin proposed for the Kolmogorov type complexity.

Def. 2.2.2.[7] x is <u>C'-random</u> iff $\exists c \forall n[K(x /n) > n - 3|n| - c]$.

THEOREM 2.2.1. If x is C-random then x is C'-random.

Proof: If x is C-random then:

$$\exists c \forall n[H(x^n) > n - c].$$

However $\exists c \forall n[H(x^n) = H(x^n/n) + H(n) + c]$ (Theorem 1.3.8.a) and therefore x is C-random implies

$$\exists c \forall n[H(x^n/n) > n - H(n) - c].$$

However since there exist a c such that for all n:

$\tilde{H}(x^n/n) \geq H(x^n/n) - c$ (Th. 1.4.4) and

$H(n) < (1 + \varepsilon)|n| + c$ (Th. 1.3.7.c)

we have that if x is C-random:

(*) $\exists c \forall n[\tilde{H}(x^n/n) \geq n - (1 + \varepsilon)|n| - c]$.

By Theorem 1.4.6(b) we have:

$$\exists c \forall n[\tilde{H}(x/n) \leq K(x^n/n) + (1 + \varepsilon)|K(x^n/n)| + c]$$

which together with the fact that $\exists c \forall n[K(x^n/n) < n + c]$ implies that:

$$\exists c \forall n[\tilde{H}(x^n) \leq K(x^n/n) + (1 + \varepsilon)|n| + c].$$

Transposing we get:

$$\exists c \forall n [K(x^n/n) \geq \tilde{H}(x\ /n) - (1 + \varepsilon)|n| - c]$$

which combined with (*) above yields that if x is C-random

$$\exists c \forall n [K(x^n/n) \geq n - 2(1 + \varepsilon)|n| - c > n - 3|n| - c]$$

which by definition implies that x is C'-random.

Q.E.D.

2.3. <u>A Definition Based on Chaitin's Formulation</u>.

Because of the complexity oscillations which the theorems of Feller and Martin-Lof tell us to expect, it seems natural to attempt a definition of randomness of the form: there exist infinitely many n for which $H(x^n)$ is maximal similar to the definitions of this type given for $K(x^n)$ and $K(x^n/n)$.

Def. 2.3.1. x is <u>B-random</u> iff $\exists c \overset{\infty}{\exists} n [H(x^n) > n + H(n) - c]$.

We now consider the relationship between B-random and K-random sequences.

THEOREM 2.3.1. x is B-random implies that x is K-random.

Proof: We will assume that x is B-random but K-non random.

Negating the definition of K-randomness we get:

x is K-non random iff $\underline{\lim}_{n\to\infty}(n - K(x^n/n)) = \infty$.

However, since $K(x^n/n) \geq K(x^n/n^*) - c$ where n^* is an optimal program for n in the sense of Chaitin

(Theorem 1.4.2) we have:

$$x \text{ is K-non-random implies: } \varliminf_{n\to\infty}(n - K(x^n/n^*)) = \infty.$$

On the other hand since x is B-random for infinitely many n there exists a c such that:

$$H(x^n) = H(x^n/n) + H(n) + c > n + H(n) - c$$

and therefore:

$$(*) \qquad \varliminf_{n\to\infty}(n - H(x^n/n)) < \infty$$

Now let p_n be the minimal program for x^n given n^* in the sense of Kolmogorov. Then $|p_n| = K(x^n/n^*)$ and $n - |p_n| = n - K(x^n/n^*)$.

We claim there exists a computer C and a constant c such that for all n:

$$H_C(x^n/n) \le K(x^n/n^*) + 2|n - K(x^n/n^*)| + c.$$

The construction of C is similar to the construction of the computer used in Theorem 1.4.1. Note that since n^* and hence n is available to C it can determine the length of p_n by being given $n - |p_n|$. Therefore there exist constants c_1, c_2 such that for all n:

$$H(x^n/n) \le H_C(x^n/n) + c_1 \le K(x^n/n^*) + 2|n - K(x^n/n^*)| + c_2.$$

But since by assumption

$$\varliminf_{n\to\infty}(n - K(x^n/n^*)) = \infty$$

we also have:

$$\lim_{n\to\infty}(n - [K(x^n/n^*) + 2|n - K(x^n/n^*)|])$$

$$= \lim_{\to\infty}([n - K(x^n/n^*)] - 2|n - K(x^n/n^*)|)$$

$$= \infty.$$

But since $H(x^n/n) \leq K(x\ /n^*) + 2|n - K(x^n/n^*)| + c$ we conclude:

$$\lim_{n\to\infty}(n - H(x^n/n)) = \infty$$

which contradicts (*).

Q.E.D.

2.4. Recursive and Recursively Enumerable Sub-sequences.

Let $f(n)$ be a recursive function with infinite range. Let x be an arbitrary binary sequence. Then by x_f we mean the sub-sequence obtained from x by deleting from x every bit x_i for which $i \notin \text{Range}[f(n)]$.

THEOREM 2.4.1. If x is K-random and the range of $f(n)$ is recursive then x_f is K-random.

Proof: Since the range of $f(n)$ is recursive we assume without loss of generality that $f(n)$ enumerates its range in increasing order. Let $j(n)$ equal the number of elements of the range of $f(n)$ less than n. By x_{D_f} we will mean the sequence obtained from x by deleting from x all bits x_i for which $i \in \text{Range}[f(n)]$. We will now assume x_f to be K-non-random and x to be

K-random and derive a contradiction.

Let A_f be the binary encoding of a program which enumerates the range of $f(n)$.

Consider the following program p_n for x^n:

$$\overline{A_f} \cdot x_{D_f}^{n-j(n)} \cdot (x_f^{j(n)})^* .$$

Computer C operates as follows on input p_n given n:

1. Evaluate $f(1), f(2) \ldots$ iteratively until $f(k) > n$.
2. Record $f(1), f(2), f(k-1)$ and set $j(n) = k-1$.
3. Reserve n cells on which to record the output x^n.
4. Place each of the $n - j(n)$ bits following $\overline{A_f}$ successively into each cell i for which $i \not\in \text{Range}[f(n)]$.
5. Simulate U on the remaining bits of the input placing the output successively into the empty output cells.

Now if x_f is non-random then $\underline{\lim}_{n\to\infty}(j - K(x_f^j)) = \infty$.

Therefore

$$\underline{\lim}_{n\to\infty}(n - |p_n|) = \underline{\lim}_{n\to\infty}(n - [c + (n - j(n)) + K(x_f^{j(n)})])$$

$$= \underline{\lim}_{n\to\infty}(j(n) - [K(x_f^{j(n)}) + c]) = \infty$$

which implies x is non-random.

Q.E.D.

[Note that in the above proof we used both versions of the definitions of K-randomness deriving a contradiction from the absolute non-randomness of x_f and the conditional randomness of x.].

The above theorem illustrates one example where a constant length prefix allowed us to delineate between two separate parts of a program. Of course, we made crucial use of the fact that the range of $f(n)$ was recursive. If the range of $f(n)$ is not recursive we can by an argument similar to the above conclude:

$$\exists c \overset{\infty}{\exists} n [K(x_f^n) > n - (1 + \varepsilon)|K(x_f^n)|]$$

since $(1 + \varepsilon)|K(x_f^n)|$ bits might be needed to delineate between the part of the program used to compute x_f^n from the part used to compute $x_{D_f}^{i-j(i)}$ where $j(i) = n$.

Chapter 3

AN ANALYSIS OF K(s*/s)

3.1. The Set of Optimal Programs.

We begin this chapter by considering the set of optimal programs. We will show that set of optimal programs is immune. Though we will assume throughout this section that we are dealing with the absolute complexity formulation of Kolmogorov, the basic theorems remain valid for all of the complexity formulations discussed in Chapter one.

Def. 3.1.1. $OP = \{p \mid \exists s[(U(p) = s) \; \& \; (\forall q < p)(U(q) \neq s]\}$

THEOREM 3.1.1. $\exists c \forall p[p \in OP \rightarrow \big| K(p) - |p| \big| < c]$ (i.e. $p \in OP$ implies p is random).

Proof: Consider a computer C defined by:

$$C(p) = U(U(p)).$$

Let $p = t^*$ and $s = p^*$. $s = p^*$ implies $\exists c \forall p[|s| \leq |p| + c]$. But since $C(s) = t$, $K_C(t) \leq |s|$ and therefore $|p| = K(t) \leq |s| + c_{C,U}$ [where $c_{C,U}$ depends only on C and U]. Hence $\big||p| - |s|\big| = \big||p| - K(p)\big| \leq \max\{c, c_{C,U}\}$.

Q.E.D.

THEOREM 3.1.2. a) OP is not recursively enumerable.

b) If S is any infinite r.e. set

then $S \cap \overline{OP}$ is infinite (i.e. OP is immune).

Proof: OP is infinite since $s^* = t^*$ implies $s = t$ and therefore b) implies a). Hence we need only prove b).

Let F be an algorithm that enumerates S without repitition, i.e. $S = \{F(0), F(1), \ldots, F(n), \ldots\}$.

Define $G(n)$ by:

$$G(0) = F(0)$$

$$G(n+1) = F[\mu j[|F(j)| > \max\{n + 1, |G(n)|\}]].$$

<u>Claim 1</u>. G enumerates a set S' without repitition.

<u>Claim 2</u>. $S' \subset S$.

<u>Claim 3</u>. S' is infinite.

Proof: Since the range of $F(j)$ is infinite there will always be a value of j for which $|F(j)|$ is greater than $\max\{n + 1, |G(n)|\}$.

<u>Claim 4</u>. If $s = G(i)$ for any i then $|s| = |g(i)| > i$.

<u>Claim 5</u>. $K_G(s) = K_G(G(i)) \leq |i|$.

<u>Claim 6</u>. $K(s) \leq K_G(s) + c_{G,U} \leq |i| + c$.

Hence for any $s \in S'$ $K(s) \leq ||s|| + c$ which implies that all but a finite number of the elements of S' are also in $\overline{OP}$.

Q.E.D.

3.2. <u>K(s*/s)</u>.

A major difference between the Kolmogorov and Chaitin formulations arises when we consider $K(s^*/s)$ and

$H(s^*/s)$.

By Theorem 1.3.4.c $H(s^*/s) \leq c$ where c is independent of s.

However $K(x^*/x)$ behaves quite differently. First we state a somewhat obvious result for $K(s^{**}/s)$.

THEOREM 3.2.1. If $||s| - |s^{**}|| = c'$ then $\exists c \forall s[K(s^{**}/s) < |c'| + c]$.

Proof: Define a computer C as follows: Given s and $0c'(1c')$ as input, C iteratively simulates k steps $[k = 1,2,\ldots,n,\ldots]$ of the computation of U on each of the $2^{|s|-c'}(2^{|s|+c'})$ sequences of length $|s| - c'(|s| + c')$ until it finds an input on which U outputs s in the least number of steps. If it finds more than one input which outputs s in the minimum number of steps it chooses the first with respect to the ordering of X^*. Hence

$$K_C(s^{**}/s) \leq 1 + |c'|$$

which implies

$$K(s^{**}/s) \leq |c'| + c.$$

Q.E.D.

COROLLARY: $K(s^{**}/s) \leq \min\{|s^{**}|, c'\} + c$.

Note that for s^* even a result as weak as the above may not be true, since one could not guarantee that having found a program p such that $|p| = K(s)$ and $U(p) = s$ that there does not exist another program p' such that $|p'| = |p|$, $p' < p$ and $U(p') = s$.

An unproved conjecture[1] is that there exists a c such that infinitely often $K(s^*/s) > |s^*| - c$.

We now prove a weaker result in the direction of the conjecture.

THEOREM 3.2.1. $\sim\exists c \forall s[K(s^*/s) \leq c]$.

Proof: Assume such a c existed. Let $r = 2^{c+1}$ and let $c_0, c_1, \ldots, c_r$ be a listing of all strings of length less than or equal to c. Then for every s there is a c_i such that:

$$U(c_i, s) = s^*.$$

We now non-constructively construct an algorithm which will output only optimal programs, hence contradicting the immunity of the set OP.

Define $\#(s)$ as the number of c_i's, $0 \leq i \leq r$ for which both $U(c_i, s)$ and $U(U(c_i, s)$ halt. Let $k = \lim\sup(\#(s)$. Then: 1) there exist s' such that for all $s > s'$, $\#(s) \leq k$, 2) there exist infinitely many s such that $\#(s) = k$.

We now provide an algorithm A which

1) is undefined if either $s \leq s'$ or $\#(s) < k$

2) outputs s^* if $\#(s) = k$ and $s > s'$.

A computes as follows:

1) If $s \leq s'$ $A(s)$ is undefined

2) If $s > s'$ dovetail the computation of

$U(U(c_i,s)$ for $0 \leq i \leq r$.

3) If and when $U(U(c_i,s))$ has halted for k different values of c_i list each $U(c_i,s)$ for which $U(U(c_i,s) = s$. Determine s^* by finding the first element in lexocagraphic order on the list. Print s^*.

A dovetailing of the computation of $A(s)$ provides an algorithm which outputs an infinite list of optimal programs.

Q.E.D.

Chapter 4

ω_U - THE PROBABILITY THAT A UNIVERSAL COMPUTER IN THE SENSE OF CHAITIN HALTS

In Chapter 1 we defined $\omega = \omega_U = \sum_s P(s) = \sum_{U(p) \text{ defined}} \frac{1}{2^{|p|}}$. We now show that in one sense it can be said that a given universal computer is preferable to another in the sense that $H_U(s) < H_{U'}(s)$. This is accomplished by showing how one can, given any universal computer U, find another which while being defined on the same number of inputs chooses those inputs to be shorter and hence has a higher probability of halting.

THEOREM: Let U be a particular universal computer for which $1 - \frac{1}{2^{k-1}} < \omega_U < 1 - \frac{1}{2^k}$. Then there exist another universal computer U' such that

1. $H_{U'}(s) \leq H_U(s)$
2. For all p sufficiently large $U(p) = s$ implies $\exists p'[(|p'| < |p|) \;\&\; U'(p') = s]$ and hence,
3. For all s sufficiently large $H_{U'}(s) < H_U(s)$.
4. $\omega_{U'} > 1 - \frac{1}{2^k}$.

Proof: We first prove (1) and (2). Let A be an algorithm which enumerates $<s_i, n_i>$ where $s_1, s_2, \ldots$

is a list of all outputs of U and n_i is the length of the program which gives output s_i.

$$\omega_U = \sum_{i=1}^{\infty} \frac{1}{2^{n_i}}$$

Since $\omega_U > 1 - \frac{1}{2^{k-1}}$ $\exists i_0$ such that:

$$(*) \qquad \sum_{i=1}^{i_0} \frac{1}{2^{n_i}} > 1 - \frac{1}{2^{k-1}}.$$

Clearly if $U(p)$ is defined and $<U(p), |p|>$ is output by A after $<x_{i_0}, n_{i_0}>$ then $|p| > k$. Otherwise

$$\omega_U = \sum_{i=1}^{\infty} \frac{1}{2^{n_i}} > \sum_{i=1}^{0} \frac{1}{2^{n_i}} + \frac{1}{2^{|p|}} > \left(1 - \frac{1}{2^{k-1}}\right) + \frac{1}{2^k} = 1 - \frac{1}{2^k}$$

Let $m = \max\{n_i | i \leq i_0\}$. Define an algorithm A' which enumerates

$$<s_i', n_i'> = \begin{cases} <s_i, n_i> & i \leq i_0 \\ <s_i, n_i - 1> & i > i_0 \end{cases}$$

Claim: $\sum_{i=1}^{\infty} \frac{1}{2^{n_i'}} < 1.$

Proof:

$$\sum_{i=1}^{\infty} \frac{1}{2^{n_i'}} = \sum_{i=1}^{i_0} \frac{1}{2^{n_i}} + \sum_{i=i_0+1}^{\infty} \frac{1}{2^{n_i'}} = \sum_{i=1}^{i_0} \frac{1}{2^{n_i}} + 2 \sum_{i=i_0+1}^{\infty} \frac{1}{2^{n_i}}$$

$$= \sum_{i=1}^{i_0} \frac{1}{2^{n_i}} + 2\left(\omega_u - \sum_{i=1}^{i_0} \frac{1}{2^{n_i}}\right)$$

$$= 2\omega_u - \sum_{i=1}^{i_0} \frac{1}{2^{n_i}} < 2\left(1 - \frac{1}{2^k}\right) - \sum_{i=1}^{i_0} \frac{1}{2^{n_i}}$$

$$= 2 - \frac{1}{2^{k-1}} - \sum_{i=1}^{i_0} \frac{1}{2^{n_i}} = 1 + \left(\left[1 - \frac{1}{2^{k-1}}\right] - \sum_{i=1}^{i_0} \frac{1}{2^{n_i}}\right)$$

$< 1.$

So by Theorem 1.3.6 there is a universal computer satisfying the set of requirements generated by A'.

This proves (1) and (2) with "all p sufficiently large" in the statement of (2) replaced by "all $p \ni |p| > m$"

(3) follows immediately from (2).

(4) To prove (4) consider 2 possibilities:

a) If $\omega_{U'} > 1 - \frac{1}{2^k}$ then we are through.

b) Otherwise we iterate the process. Clearly U' satisfies the hypotheses of the theorem. Let A' replace A as the algorithm to enumerate all pairs $<s_i', n_i'>$ outputted by A'. Note that

$$\omega_{U'} = \sum_{i=1}^{i_0} \frac{1}{2^{n_i}} + 2\left[\omega_U - \sum_{i=1}^{i_0} \frac{1}{2^{n_i}}\right].$$

Hence the process iterated gives:

$$\omega_{U''} = \sum_{i=1}^{i_0} \frac{1}{2^{n_i}} + 4\left[\omega_U - \sum_{i=1}^{i_0} \frac{1}{2^{n_i}}\right]$$

If $\omega_{U''} > 1 - \frac{1}{2^k}$ we are done otherwise

$$\omega_{U'''} = \sum_{i=1}^{i_0} \frac{1}{2^{n_i}} + 8\left[\omega_U - \sum_{i=1}^{i_0} \frac{1}{2^{n_i}}\right].$$

So that eventually

$$\omega_{U\overbrace{''''\cdot\cdot'}^{n}} = \sum_{i=1}^{i_0} \frac{1}{2^{n_i}} + 2^n\left[\omega_U - \sum_{i=1}^{i_0} \frac{1}{2^{n_i}}\right]$$

will exceed $1 - \frac{1}{2^k}$.

Q.E.D.

COROLLARY: $\limsup \omega_{U_i} = 1.$

Proof: By (4) of the previous theorem we can come as close to 1 as we like.

The above result implies that there are no optimal universal computers in the strict sense.

Chapter 5

SUBRECURSIVE FORMULATIONS OF DESCRIPTIVE COMPLEXITY

5.1. Introduction and Fundamental Definitions

We now wish to consider the notion of descriptive complexity in a subrecursive formalism. Though our discussion is carried out entirely in the context of the Grzegorczyk hierarchy,[1] our result will be valid in any hierarchy which shares certain fundamental properties with the Gyzegorczyk hierarchy. We first give a listing of those properties.

To briefly outline the structure of the Gyzegorczyk hierarchy consider the sequence of functions:

$$f_0(x,y) = y + 1$$

$$f_1(x,y) = x + y$$

$$f_2(x,y) = (x+1)(y+1)$$

$$\left.\begin{array}{l} f_{n+1}(0,y) = f_n(y+1,\ y+1) \\ f_{n+1}(x+1,y) = f_{n+1}(x, f_{n+1}(x,y)) \end{array}\right\} \quad n \geq 2.$$

For $n = 1,2, \ldots,$ ξ^n is smallest class of functions such that:

(1) ξ^n contains $S(x) = x+1$, $I_1(x,y) = x$, $I_2(x,y) = y$, and $f_n(x,y)$.

(2) ξ^n is closed under composition and limited recursion. [A class $\mathcal{A}$ is closed under the operation of limited recursion if for any functions g,h,i which are in class $\mathcal{A}$ the function f defined by:

$$f(x,o) = g(x)$$
$$f(x,y+1) = h(x,y,f(x,y)),$$

and satisfying:

$$f(x,y) \leq i(x,y)$$

is also in class $\mathcal{A}$.]

We note the following results concerning the hierarchy:

1) ξ^n is closed under bounded minimalization.[1]

2) $\xi^n \subset \xi^{n+1}$ [1]

3) ξ^3 is the class of elementary functions[1]

4) For $n > 2$, ξ^{n+1} contains a universal function for the one-place functions of class ξ^n.[1]

5) $f_{n+1}(x,x)$ majorizes all one-place functions of level ξ^n in the sense that given $f \in \xi^n$ $\exists c \forall x[f_{n+1}(x+c,x+c) > f(x)]$ for all x.[1]

6) There exists an infinite chain of classes of algorithms written in a suitable formal language which we denote by E^1, E^2, E^3 ... with the following properties:

a) If f is a function in ξ^i then there is an algorithm $A \varepsilon E^i$ which computes f.

b) For any algorithm $A \varepsilon E^i$ the function computed by A is in Grzegorczyk class ξ^i.

c) Given any algorithm A and integer i it is decidable whether $A \varepsilon E^i$.

7) $\bigcup_{n=1}^{\infty} \xi^n$ = PR, the class of primitive recursive function[1] and hence $\bigcup_{n=1}^{\infty} E^n$ contains a class of algorithms for the primitive recursive functions.[2]

8) For any recursive function $g(x)$, if $g(x) \leq f(x)$ and $f(x) \varepsilon \xi^i$ then $g(x) \varepsilon \xi^i$.[1]

It is important to note that many algorithmic formalisms (e.g. Turing machines or defining equations) do not satisfy 6) c) above. However Minsky and others[2] exhibit algorithmic formalisms based on simplified programming languages which contain:

1). A class of elementary instructions such as successor and identity.

2) Conditional forward transfers.

3) Looping instructions in which the parameters governing the number of times the statements in the scope of the loop are to be executed cannot be changed by any statement within the scope of the loop.

4) Conditional backward transfer.

Programming languages allowing for all four types of instructions provide formalisms for the full class of partial recursive functions while programming languages without conditional backward transfer and restrictions on the depth of nesting of loop instructions provide a formalism for the Grzegorczyk hierarchy which satisfies the decidability requirement of 6) c) above.[2]

Let $b(n)$: $N \to X^*$ be the function which maps n into the nth element of X^*. Let $b^{-1}(n)$: $X^* \to N$ be its inverse with B and B^{-1} algorithms for $b(n)$ and $b^{-1}(n)$ respectively.

Let $A'_0, A'_1, \ldots, A'_n$ be a list of algorithms for all partial recursive functions: $N \to N$. If $A_i = BA'_iB^{-1}$ then $A_0, A_1, \ldots, A_n, \ldots$ is a list of all algorithms for partial recursive functions: $X^* \to X^*$.

Def. 5.1.1. $\underline{A_i \varepsilon E^j}$ iff $A'_i \varepsilon E^j$.

Note that "$A_i \varepsilon E^j$" is decidable since "$A'_i \varepsilon E^j$" is decidable by requirement 6) c).

Relative to a particular listing of all algorithms $A_0, A_1, A_2, \ldots, A_n, \ldots$ we define:

$$U(o^i 1p) = A_i(p)$$

as the universal computer on which our treatment is based.

We now define a universal optimal computer for each level $j \geq 3$ of the hierarchy. [All results in this chapter are claimed only for levels ξ^i, $i \geq 3$.].

Def. 5.1.2. $U^j(o^i 1p) = \begin{cases} A_i(p) & \text{if } A_i \varepsilon E \\ 0 & \text{otherwise} \end{cases}$

For functions of two variables we can similarly define a jth level universal optimal computer

Def. 5.1.3. $U^j(o^i 1p,t) = \begin{cases} A_i(p,t) & \text{if } A_i \varepsilon E \\ 0 & \text{otherwise} \end{cases}$

In terms of the universal optimal computers of one and two variables we define the notion of the descriptive complexity of s relative to level j of the hierarchy. Note that since the identity computer is present at every level of the hierarchy the complexity of any string s is never infinite.

Def. 5.1.4.

a) $K^j(s) = \min |p| [U^j(p) = s]$

b) $K^j(s/t) = \min |p| [U^j(p,t) = s]$

c) $H^j(s) = \min |p| [U^j(p) = s$ where p is self delimiting as previously explained]

d) $s^{*j} = \min p[U^j(p) = s]$. Note that s^{*j} will be used to refer to the minimal program for s in either the Kolmogorov or Chaitin formulation. This should be clear in context.

THEOREM 5.1.2. x is i-random(1) iff x is i-random(2).

As a result we will define x to be K_i-random iff x is i-random(1) or x is i-random(2).

The statement and proof of Theorem 2.3.1 carries over directly. Hence as before we can prove:

THEOREM 5.1.3. x is B_i-random $\Rightarrow$ x is K_i-random.

While the converse implication was left unsettled in Chapter two, in the next section we will show that: x is K_{i+1}-random implies x is B_i-random.

5.2. $K^i(s^{*j}/s)$ and Related Results.

We showed earlier (Theorem 3.1.2) that the set of optimal programs was immune. We now show that in a sub-recursive formalism OP_i and hence $\overline{OP_i}$ is recursive.

Def. 5.2.1. $OP_i = \{p \mid \exists t[(U^i(p) = t) \ \& \ (\forall q < p) U^i(q) \neq t]\}$.

THEOREM 5.2.1. OP_i is recursive.

Proof: Given a program p we evaluate $U^i(p)$ and then check $U^i(q)$ for every $q < p$ to see whether or not $U^i(q) = U^i(p)$. Note that $U^i(p)$ is in Grzegorczyk level ξ^{i+1} and therefore the predicate: "$p = s^{*j}$" is a relalation in Grzegorczyk level ξ^{i+1}.

Q.E.D.

COROLLARY: There exist c such that for all s

a) $K^{i+1}(s^{*i}/s) \leq c$

b) $\tilde{H}^{i+1}(s^{*i}/s) \leq c$.

THEOREM 5.2.2. x is K_{i+1}-random implies that x is B_i-random.

Proof: Clearly $\exists c \forall n [K^{i+1}(x^n/n) \leq \tilde{H}^{i+1}(x^n/n) + c]$. By the above corollary n^{*i} can be obtained from n by a fixed length program and therefore:

$$\exists c \forall n [\tilde{H}^{i+1}(x^n/n) \leq H^i(x^n/n) + c].$$

Hence:

$$\exists c \forall n [K^{i+1}(x^n/n) \leq H^i(x^n/n) + c].$$

Now assume that x is K_{i+1}-random but not B_i-random. If x is B_i-non-random then

$$\underline{\lim}_{n\to\infty}(n - H^i(x^n/n)) = \infty$$

but

$$\exists c \forall n [K^{i+1}(x^n/n) \leq H^i(x^n/n) + c]$$

implies that

$$\underline{\lim}_{n\to\infty}(n - K^{i+1}(x^n/n)) = \infty$$

which contradicts the assumption that x is K^{i+1}-random

Q.E.D.

5.3. A Hierarchy of Random Sequences.

Given that x is K_i-random (B_i-random) it is clear that x is *a fortiori* K_{i-1}-random (B_{i-1}-random). It is unclear whether the converse is valid. We think not and in this section outline a construction which given a sequence x which is K-random (B-random) constructs another sequence y which is K_{i+1}-non-random (B_{i+1}-non-random) but which we conjecture is K_i-random (B_i-random). If our conjecture is proved false and it is further proved that x is K_i-random (B_i-random) implies that x is also K-random (B-random) then each of the following results would follow:

1) x is K_i-random implies that x is K_{i+1}-random

2) x is K_i-random implies that x is B_i-random

3) x is K-random iff x is B-random.

Note that:

1) is weaker than the hypothesis

2) follows from 1) and Theorem 5.2.2.

3) follows from the following chain of equivalences:

x is K-random $\Leftrightarrow$ x is K_i-random $\Leftrightarrow$ x is B_i-random $\Leftrightarrow$ x is B-random.

We feel, however, that the following conjecture is valid:

CONJECTURE: There exists a sequence y which is K_i-random (B_i-random) and is not K_{i+1}-random (B_{i+1}-random).

We give the following construction in support of the conjecture:

Let x be a random string (K-random or B-random). Let F be a function $N \to N$ such that $F \in \xi^{i+1}$ but F dominates any function $g \in \xi^i$. Define a string y as follows:

Let $n_o = F(o)$ and $y^{n_o+1} = x^{n_o}1$.

Compute $F(x^{n_o}1) = n_1$ and set

$$y^{n_1+2} = x^{n_o}1x_{n_o} \dots x_{n_1}1$$

$$= y^{n_o+1} x_{n_o+1} \dots x_{n_1}1 .$$

Continuing iteratively we let $n_k = F(y^{n_{k-1}+k})$ and set

$$y^{n_k+(k+1)} = y^{n_{k-1}+k} x_{n_{k-1}} \dots . x_{n_k}1.$$

This defines a sequence y which is clearly non-random at level $i+1$ since a program for y^n is given by p_n:

c bits to encode the function $F \cdot x^{n-j(n)}$

where $j(n)$ is a count of the number of 1's inserted in $x^{n-j(n)}$ to give y^n. Hence

$$\lim_{n\to\infty}(n - |p|) = \lim_{n\to\infty}(n - (n - j(n) + c)) = \lim_{n\to\infty}(j(n) - c) = \infty.$$

A similar argument shows that y is also B_{i+1}-non-random. We are unable to show that y is K_i-random or B_i-random and the conjecture remains unproved.

5.4. The Maximum Grzegorczyk Level of All Programs of Length Less Than n.

In this section we prove a theorem based on Blum's fundamental theorem on the size of machines (algorithms) which clearly underlines an important distinction between the Grzegorczyk hierarchy and the hierarchy of classes of algorithms discussed in section one. Let $A_0, A_1, A_2, \ldots A_n \ldots$ be an effective Gödel numbering of all algorithms in a Minsky-type programming language formalism for the set of partial recursive functions.

Def. 5.4.1.[4] A recursive function s mapping N (viewed as the set of indices) into N (viewed as the set of sizes) is called a size measure, $s(A_i)$ denoted by $|i|$ being called the size of A_i iff

1) There exist at most a finite number of machines of any given size and
2) there exists an effective procedure for deciding, for any j, which algorithms are of size j.

[Note that in this section only we will follow Blum's convention and let $|i|$ denote the size of the ith algorithm.].

THEOREM 5.4.1. (Blum's fundamental theorem on machine size).

1. Let g be any recursive function with infinite range.
2. Let f be any recursive function. Then there exist $i,j \in N$, both uniform in f,g such that:

   a) $A_i = A_{g(j)}$
   b) $f|i| < |g(j)|$.

Consider all algorithms of size n or less. By the axioms a size measure must satisfy there are at most a finite set of algorithms whose size is less than n. Hence there exists an effective procedure to determine the maximum class E^i in which any algorithm belonging to any E^j lies. Similarly there are at most a finite number of functions computed by an algorithm of size less than n. Hence among those functions computed there must exist at least one function f such that if $f \in \xi^i$ then for any other function g computed either g is not primitive recursive or $g \not\in \xi^i$. We summarize the above with the following definition:

Def. 5.4.2.

$$G(n) = \begin{cases} \max(i) & f \text{ is 1. primitive recursive, 2. computed by a algorithm of size less than } n, \text{ and 3. contained in } \xi^i. \\ 0 & \text{if no primitive recursive function is computed by a program of size less than } n. \end{cases}$$

THEOREM 5.4.2. $G(n)$ is not recursively bounded.

Proof: Assume $G(n) < h_1(n)$ for some recursive function $h_1(n)$. Define $h_2(n) = \max[h_1(n), h_1(|n|)]$.

(1) $G(n) < h_2(n)$.

For every k define $h_3(k) = \max[h_2(\ell_1), \ldots, h_2(\ell_m)] + 1$ where $\ell_1, \ell_2, \ldots, \ell_m$ are the indices of all programs of size k.

(2) $h_3(|n|) > h_2(n) > G(n)$.

(3) $\forall i,j$ if $\varphi_i \in \xi_j$ and $|i| = k$ then

then $h_2(i) > h_1(|i|) = h_1(k) > G(k) > j$.

(4) Putting (2) and (3) together we get

$h_3|i| > j$ where $\varphi_i \in \xi_j$.

Now let $g(n)$ enumerate the canonical indices of f_n where f_n is the intial function added at level ξ_n. $g(n)$ is strictly monotone. Define $\ell(n) = |g(n)|$. $\ell(n)$ is likewise a strictly monotone recursive function. Finally $h_4 = \ell \circ h_3$.

We now apply Blum's theorem to h_4 and g:

Given g a recursive function with infinite range and h_4 as above there exist i,j such that

a) $\varphi_i(x) = \varphi_{g(j)}(x)$.

b) $h_4(|i|) < |g(j)| = \ell(j)$.

Combining b) and (4) we have

(4) $h_3|i| > j$ since $\rho_i = \rho_{g(j)} \varepsilon \xi_j$

b) $\ell(h_3(|i|)) < \ell(j)$.

Since ℓ is monotone we can remove ℓ from both sides of the inequality giving:

b') $h_3(|i|) < j$ which is the desired contradiction.

Q.E.D.

5.5. The Longest String of Complexity Less Than n.

While results in this section are stated in terms of $K(s)$ and $K^i(s)$, analogous results hold for $H(s)$ and $H^i(s)$. For the remainder of this section we denote the string associated with n by n'.

Let A be an algorithm for a function $f: X^* \to X^*$.

Def. 5.5.1. $K(A) = \min |s|[U(s)$ is a binary encoding of $A]$.

Chaitin has defined[5] two functions which are analogous to Rado's[6] "busy beaver" functions.

Def. 5.5.2. $a(n') = \max s[K(s) \leq n]$.

Def. 5.5.3. $b(n') = \max A(n')[K(A) \leq n]$

[Note that $a(n')$ & $b(n')$ may not be defined for small values of n' since there is no guarantee that the set of s for which $K(s) \leq n$ or the set of A for which $K(A) \leq n$ is non-empty.].

Chaitin has shown:[7]

THEOREM 5.5.1. a) $\exists c \forall n [a((n+c)') > b(n') \ \& \ b((n+c)') > a(n')]$.

b) $a(n')$ and hence $b(n')$ dominate all recursive functions.

We now analogously define $a_i(n')$ and $b_i(n')$.

Def. 5.3.4. $a_i(n') = \max s(K^i(s) \leq n)$.

Def. 5.3.5. $b_i(n') = \max A(n')[K^i(A) \leq n \ \& \ A \ \varepsilon \ E^i]$

We now wish to show that $a_i(n') \ \varepsilon \ \xi^{i+1}$ but not in ξ^i.

THEOREM 5.5.2. $a_i(n') \not\varepsilon \ \xi^i$.

Proof: Assume $a_i(n') \ \varepsilon \ \xi^i$, then $a^*(n') = 1 \cdot a_i(n')$ is also in ξ^i.

Let A_k be an algorithm for $a_i^*(n')$. Consider $U^i(0^k\,1\,1^{n-(k+1)})$ for $n > k+1$.

$$|0^k\,1\,1^{n-(k+1)}| = n.$$

$U^i(0^k\,1\,1^{n-(k+1)}) = A\ (1^{n-(k+1)}) = a_i^*(1^{n-(k+1)})$

$= 1 \cdot a_i(1^{n-(k+1)})$. Since the string $1^{n-(k+1)}$ corresponds to the integer $2^{n-k} - 2$

$$U^i(0^k\,1\,1^{n-(k+1)}) = 1 \cdot a\ (1^{n-(k+1)}) > a\ (n').$$

which contradicts the definition of $a_i(n)$.

Q.E.D.

THEOREM: $a^i(n') \varepsilon \xi^{i+1}$.

Proof: We show

1) $b_i(n') \varepsilon \xi^{i+1}$

2) $\exists c \forall n[a_i(n') \leq b_i((n+c)')]$.

Proof of 1):

Consider the following algorithm for $b_i(n')$.

1. Compute 2^{n+1}.
2. Check each of the 2^{n+1} sequences t of length less than or equal to n to see if $U^i(t)$ is a binary encoding of a program in ξ^i.
3. For each t for which $U(t)$ is the binary encoding of an algorithm $A \varepsilon \xi^i$ compute $A(n')$.
4. Output the maximum $A(n')$.

Clearly each of the above steps can be performed in ξ^{i+1}.

Proof of 2):

Consider $A_{i,n'}$ an algorithm for the constant function whose value is always $a_i(n')$.

Since $K^i(A_{i,n'}) \leq K^i(a_i(n)) + c \leq n + c$ we can conclude that

$$a(n') = A_{i,n'}((n+c)') \leq \max A(n')[A(n') \in \xi^{i} \; \& \; K^{i}(A) \leq n+c] = b((n+c)').$$

Q.E.D.

NOTES

Chapter 1.

1. The fundamental papers on which most of this section is based are Chaitin [4] and [5], Kolmogorov [13] and [14], Solomonoff [22], and Martin-Löf [17]. For a more complete introduction see Zvonkin and Levin [24].

2. See Chaitin [5] and [6] and Kolmogorov [13].

3. See Kolmogorov [13] and Solomonoff [22].

4. See Martin-Löf [17].

5. See Blum [2] for an abstract definition of a complexity measure.

6. See Kolmogorov [13].

7. See Martin-Löf [17].

8. See Kolmogorov [13].

9. See Chaitin [7].

10. See Chaitin [7], pp. 5-6.

11. For a complete discussion of the correspondence see Zvonkin and Levin [24], sections, Willis [23], and Chaitin [7].

12. The remaining definitions and theorems of this section are all from Chaitin [7].

13. The extent to which $P(s)$ and $P(s/t)$ are independent of the particular universal computer U is not explicitly discussed. However, Theorem 3.7 in Chaitin [7] and Chapter 4 in this paper shed some light on the situation.

14. See Loveland [16] where this theorem and its converse are proven.

15. See Barzadin [1], Theorems 1 and 2.

Chapter 2.

1. See Feller [11], p. 210, problem 5.
2. See Martin-Löf [18] and Zvonkin and Levin [24], Theorem 2.6.
3. See Loveland [16].
4. See Daley [9].
5. See Martin-Löf [18].
6. See Chaitin [7].
7. See Chaitin [5].

Chapter 3.

1. This conjecture was related in a conversation with G. Chaitin.

Chapter 5.

1. See Grzegorczyk [12] where all properties of the Grzegorczyk hierarchy stated in this section are proven.
2. For a detailed formulation of a subrecursive formalism with the required properties see Minsky [20], Chapter 11, Meyer and Ritchie [19], and Constable [8].
3. See Daley [9].
4. See Blum [2].
5. See Chaitin [6], Section 5.
6. See Lin and Rado [15].
7. See Chaitin [6], Section 5.

BIBLIOGRAPHY

[1] Barzdin, Y.M., "The complexity of programs to determine whether natural numbers not greater than n belong to a recursively enumerable set," Soviet Math. Dokl. 9 (1968), 1251-1254.

[2] Blum, M., "A machine-independent theory of the complexity of recursive functions," J. ACM 14 (1967), 322-337.

[3] Blum, M., "On the size of machines," Information and Control 11 (1967), 257-265.

[4] Chaitin, G. J., "On the length of programs for computing finite binary strings," J. ACM 13 (1966), 547-569.

[5] Chaitin, G. J., "On the length of programs for computing finite binary sequences: statistical considerations," J. ACM 16 (1969), 145-159.

[6] Chaitin, G. J., "Information-theoretic limitations of formal systems," J. ACM 21 (1974).

[7] Chaitin, G. J., "A theory of program size formally identical to information theory," to be published as an I.B.M. Research Report.

[8] Constable, R. L., "Subrecursive programming languages II on program size," Journal of Computer and Systems Sciences 5 (1971), 315-334.

[9] Daley, R. P., "Complexity and Randomness," Computational Complexity Symposium 7, Courant Institute (1973), 113-122.

[10] Davis, M., Computability and Unsolvability. McGraw-Hill, New York, 1958.

[11] Feller, W., An Introduction to Probability Theory and Its Applications, 3rd ed., Wiley, New York, 1968.

[12] Grzegorczyk, A., "Some classes of recursive functions," Rozprawy Mathematicyzne (1953), 1-45.

[13] Kolmogorov, A. N., "Three approaches to the quantitative definition of information," Problems of Information Transmission 1 (1965), 1-7.

[14] Kolmogorov, A. N., "On the logical foundations of information theory and probability theory," IEEE Transactinon Information Theory 14 (1968), 662-664.

[15] Lin, S. and T. Rado, "Computer studies of Turing machine problems," ACM 12 (1965), 196-212.

[16] Loveland, D. W., "A variant of the Kolmogorov concept of complexity," Information and Control 15 (1969), 510-526.

[17] Martin-Löf, P., "The definition of random sequences," Information and Control 9 (1966), 602-619.

[18] Martin-Löf, P., "Complexity oscillations in infinite binary sequences," Zeitschrift fur Wahrscheinlichkeitstheorie und Verwandte Gebiete, 19 (1971), 225-230.

[19] Meyer, A. R. and D. M. Ritchie, "The complexity of loop programs," Proc. 22 National ACM Conf. (1967), 465-470.

[20] Minsky, M., Computation, Finite and Infinite. Prentice-Hall, Englewood Cliffs, N.J., 1965.

[21] Schnorr, C. P., "A unified approach to the definition of random sequences," Mathematical Systems Theory 5 (1971), 246-258.

[22] Solomonoff, R. J., "A formal theory of inductive inference, Part I," Information and Control 7 (1964), 1-22.

[23] Willis, D. G., "Computational complexity and probability constructions," ACM Journal 17 (1970), 241-259.

[24] Zvonkin, A. K and A. L. Levin, "The complexity of finite objects and the development of the concepts of information and randomness by means of the theory of algorithms," Russian Mathematical Surveys 25 (1970), 83-124.